
🔧 Conceptual Overview:

🕹️💻 Game Objective:

- The computer generates a **random number (1–100)**.
- The player gets **10 attempts** to guess the number.
- After each guess, feedback is shown whether it's too high, too low, or correct.
- If guessed correctly, the number of attempts is recorded.
- The best (lowest) number of attempts is saved as **High Score** using a text file.
- A **Reset** button restarts the game.

📚 Java Concepts & Libraries Used:

✅ Java Swing Library:

- For GUI development:
  - `JFrame`, `JPanel`, `JLabel`, `JButton`, `JTextField`, `JOptionPane`
  - Layout management via `BoxLayout`, `Box.createRigidArea`
  - Custom colors and fonts via `Font`, `Color`, and `setBackground()`

✅ AWT Library:

- For GUI painting and layout:
  - `Graphics`, `Image`, `Font`, `Component`, `Dimension`

✅ Event Handling:

- **`ActionListener` interface**: Used to handle button clicks and enter key actions.

✅ I/O (File Handling):

- For saving/loading high scores:
  - `BufferedReader`, `BufferedWriter`, `FileReader`, `FileWriter`
  - Handles **persistence** of game data across sessions using `highscore.txt`.

✅ Random Number Generation:

- `java.util.Random` class is used to generate a random number from 1 to 100.

🔁 OOP (Object-Oriented Programming) Principles Used:

✅ Encapsulation:

- Game logic and GUI components are encapsulated within the `NumberGuessGame` class.
- Variables like `randomnumber`, `attemptsleft`, and `highscore` are `private`.

✅ Abstraction:

- The user interacts with a simple GUI and does not need to understand internal logic like file handling or random number generation.

✅ Inheritance:

- The `NumberGuessGame` class **extends `JFrame`**, inheriting window behavior from Java's Swing framework.

✅ Polymorphism:

- Implementing the `ActionListener` interface and **overriding `actionPerformed()`** method.
- Also, **`paintComponent()`** in `backgroundpanel` is an overridden method for custom drawing.

✅ Composition:

- The `NumberGuessGame` class uses other objects like `JPanel`, `JButton`, `JLabel`, etc., to build the complete GUI.

🧱 Classes & Structure:

🔹 **`NumberGuessGame`** (Main class):

- Inherits from `JFrame`.
- Implements `ActionListener` to respond to user actions.
- Handles:
  - Random number generation
  - Attempt tracking
  - Highscore logic

- GUI layout

◆ **`backgroundpanel` (Inner class):**

- Inherits from `JPanel`.
- Overrides `paintComponent(Graphics g)` to paint a background image.
- Loads image from resources using `getResource("areeba.png")`.

🖼️ GUI Elements:

Element	Description
`JTextField`	To input the guessed number.
`JButton` (Guess)	To submit a guess.
`JButton` (Reset)	To reset the game after it's over.
`JLabel` (Messages)	To guide the user and give feedback.
`JLabel` (Highscore)	To show the best score.
`JLabel` (Credits)	To show developer's name.

🔄 Game Logic Flow:

1. **Start Game** → Generate random number (1-100) and set 10 attempts.
2. **User Inputs a Guess** → Evaluate guess.
   - If correct: Display success and attempts used.
   - If incorrect: Show message (too high/low), reduce attempts.
   - If out of attempts: Reveal the number.
3. **High Score** → If new score is better, save it to a file.
4. **Reset** → Generate new number, reset attempts and fields.

📁 Files Used:

- **`NumberGuessGame.java`** – Main Java source file.
- **`highscore.txt`** – Text file that stores the best score.
- **`areeba.png`** – Background image used in the game.

🔑 Important Features:

Feature	Implementation
Persistent High Score	File I/O via `BufferedReader/Writer`
Custom Background Image	Drawn using `paintComponent()`
User-friendly Messages	`JLabel` updates and `JOptionPane` pop-ups
Styled GUI	Fonts, colors, emojis used in GUI elements
Error Handling	For number parsing and file I/O
Reset Functionality	Allows starting a new game after one ends
Input Validation	Checks for valid numbers (1–100)

💡 Possible Enhancements:

- Add **sound effects** (win/lose).
- Add **timer** to track how fast a user guesses.
- Keep a **history of guesses**.
- Allow user to choose **difficulty level** (adjust number range or attempts).
- Save **player name** along with score.
- Deploy as a `.jar` file for desktop use.

✅ Summary:

Aspect	Description
Language	Java
Type	GUI-based desktop game
GUI Library	Swing + AWT
Concepts Used	OOP (Inheritance, Polymorphism, Encapsulation), File I/O, Event Handling, Random Numbers
File Structure	Single Java file + image + text file
Audience	Beginners learning GUI and OOP in Java
